

# Ateneo de Manila University

## IPC: Signals



Department of Information Systems and  
Computer Science

S.Y. 2001-2002

<http://sysads.ateneo.net/wyu/>

[wyy@admu.edu.ph](mailto:wyy@admu.edu.ph)

## Interprocess Communications

- ★ mechanism in which processes communicate with one another
- ★ in order to facilitate communication among different processes
- ★ special mechanisms for process communications:
  - signals
  - semaphores
  - shared memory
  - pipes
  - message passing

## Signals

- ★ are software generated interrupts
- ★ can also be generated when an error occurs such as SIGFPE (floating point exception) and SIGSEGV (segfault)
- ★ can also be generated when a hardware event such as a bus error or an illegal instruction is encountered
- ★ are pre-specified by the system and defined in `signal.h`

## Classes of Signals

- ★ Hardware
- ★ Software
- ★ Input/output
- ★ Process control
- ★ Resource control

## Signals

Behavior when a signal is received by a process:

- ★ signal is discarded after being received
- ★ process is terminated after the signal is received
- ★ a core file is written, then the process is terminated
- ★ process is suspended after the signal is received

## Common Signals

- ★ are number from 1 to 31
- ★ common signals defined in `signal.h`
  - SIGHUP 1 hangup
  - SIGINT 2 interrupt
  - SIGQUIT 3 quit
  - SIGILL 4 illegal instruction
  - SIGABRT 6 used by abort
  - SIGKILL 9 hard kill
  - SIGALRM 14 alarm clock
  - SIGTERM 15 soft kill
  - SIGCONT 19 continue a stopped process
  - SIGCHLD 20 to parent on child stop or exit

## Sending Signals

```
★ int kill(pid_t pid, int sig);
```

```
★ int raise (int sig);
```

- sends `sig` signal to the processes `pid`
- if `pid` is zero then it refers to all processes in the process group
- if `pid` equals -1, then `sig` is sent to every process except for the first one
- if `pid` is less than -1, then `sig` is sent to every process in the process group
- if `sig` is zero no signal is sent
- `raise ()` is equivalent to `kill(getpid(), sig)`
- returns -1 when an error occurs

## Signal Handling

- ★ signals can be caught and its behavior can be changed
- ★ what can happen after catching a signal?
  - process can let the default action happen
  - process can block the signal (some signals cannot be ignored)
  - process can catch the signal with a handler
- ★ Signal handlers
  - usually execute on the current stack of the process
  - can be changed on a per-signal basis
  - a process must resume in a different context than the interrupted one, it must restore the previous context itself

## Signal Handling

```
★ void (*signal(int signum, void  
(*sighandler)(int)))(int);
```

- installs a new signal handler `sighandler` for the signal with number `signum`
- `sighandler` can access either of the following:
  - \* `SIG_IGN` causes the signal to be ignored
  - \* `SIG_DFL` causes the signal to execute its default action
  - \* a pointer to a function
- signals `SIGKILL` and `SIGSTOP` cannot be caught or ignored

## Default Signal Actions

Signal	Number	Action	Description
SIGHUP	1	A	death of controlling process/terminal
SIGINT	2	A	interrupt from keyboard (ctrl-c)
SIGQUIT	3	C	quit from keyboard
SIGILL	4	C	illegal instruction
SIGABRT	6	C	abort signal from abort(3)
SIGFPE	8	C	floating point exception
SIGKILL	9	AEF	kill signal
SIGSEGV	11	C	invalid memory reference
SIGPIPE	13	A	broken pipe

Figure 1: Default Signal Actions

Signal	Number	Action	Description
SIGALRM	14	A	timer signal from alarm(2)
SIGTERM	15	A	termination signal
SIGUSR1	30,10,16	A	User-defined signal 1
SIGUSR2	31,12,17	A	User-defined signal 2
SIGCHLD	20,17,18	B	child stopped or terminated
SIGCONT	19,18,25		continue if stopped
SIGSTOP	17,19,23	DEF	stop process
SIGTSTP	18,20,24	D	stop typed at tty
SIGTTIN	21,21,26	D	tty input for background process
SIGTTOU	22,22,27	D	tty output for background process

Figure 2: Default Signal Actions

Signal Actions:

- A** terminate process
- B** ignore signal
- C** terminate process and dump core
- D** stop/suspend the process
- E** signal cannot be caught
- F** signal cannot be ignored



Copyright © 2000-2001 by William Emmanuel S. Yu. This material may be distributed only subject to the terms and conditions set forth in the Open Content License, v1.0 or later (the latest version is presently available at <http://opencontent.org/opl.shtml>).